VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2018 Ondřej Cibulka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ondřej Cibulka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Digital Wizards Group s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Konzultant bakalářské práce: Bc. Aleš Rosa

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5.4.2018 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.

V Ostravě 5.4.2018

Rád bych poděkoval firmě Digital Wizards Group s.r.o. za možnost vykonávat bakalářskou praxi, konkrétně Bc. Alešovi Rosovi. Déle bych chtěl poděkovat Ing. Michalovi Čerbákovi, Ph.D. za pomoc při plnění zadaných úkolů a Ing. Michalovi Radeckému. Ph.D. za odborné rady týkající se této bakalářské práce.

Abstrakt

Tato bakalářská práce popisuje mé absolvování odborné praxe ve firmě Digital Wizards Group s.r.o., ve které jsem pracoval na pozici programátor PHP/Laravel. V první části je popsáno pracovní zaměření firmy, mé pracovní prostředí a používané technologie. Druhá část popisuje jednotlivé úkoly, kterými jsem se zabýval a jejich řešení. V poslední části jsem celkově zhodnotil odbornou praxi a mé dosažené výsledky.

Klíčová slova: odborná praxe, Digital Wizards Group, s.r.o., e-shop, webové stránky, backend, October CMS, PHP

Abstract

This bachelor thesis describes my professional practice in the Digital Wizards Group s.r.o. company, where I was working as a PHP/Laravel developer. The first part describes the business focus of the company, my working enviroment and used technologies. The second part describes the individual tasks I was working on and their solutions. In the last part I evaluated professional practice and my results.

Key Words: professional practice, Digital Wizards Group, s.r.o., e-shop, websites, backend, October CMS, PHP

Obsah

Seznam použitých zkratek a symbolů

XML	–	eXtensible Markup Language
JSON	–	JavaScript Object Notation
CMS	–	Content Management System
PHP	–	Hypertext Preprocessor
URL	–	Uniform Resource Locator
HTML	–	HyperText Markup Language

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Odbornou praxi jsem vykonával ve firmě Digital Wizards Group s.r.o. Tuto firmu jsem si zvolil z důvodu, že jsem již před nástupem na bakalářskou praxi měl zkušenosti s technologií, která se v této firmě používá při vývoji webových stránek, a ve které bych se touto cestou mohl dále zdokonalit a získat nové zkušenosti.

V první části bakalářské práce představím firmu samotnou, její zaměření a používané technologie. Popíši mé pracovní prostředí a použité nástroje pro vývoj webových stránek.

Druhá část bude věnovaná jednotlivým úkolům, které jsem během mého působení ve firmě obdržel. Popíši jejich zadání, postup implementace a problémy, se kterými jsem se u každého úkolu setkal.

V poslední části zhodnotím nové zkušenosti a vědomosti, které jsem během absolvování praxe získal a také potřebné znalosti, které mi během práce scházely.

2 Popis společnosti a pracovní pozice

2.1 Popis společnosti

Firma Digital Wizards Group s.r.o. se sídlem v Ostravě-Porubě je společnost zabývající se vývojem a realizací webových stránek a e-shopů na míru. Firma pracuje na vývoji redakčního systému Woodie2, který slouží pro správu webových stránek, jako jsou například jazykové mutace, správa uživatelských oprávnění nebo tvorba newsletterů.[1] Pro správu e-shopu firma vyvíjí redakční systém Rocketoo, který umožňuje editaci produktů, zpracování objednávek, napojení na platební brány a podobně.

Oba tyto redakční systémy pracují na open-source platformě October CMS, který je založen na nejpopulárnějším PHP frameworku Laravel.

2.2 Popis pracovní pozice

Do firmy jsem se hlásil na pozici programátora PHP/Laravel. Po kontaktování firmy jsem zaslal životopis a byl jsem pozván na krátký pohovor, po kterém jsem byl na bakalářskou praxi přijat.

Pracoval jsem v kanceláři s dalšími dvěma spolupracovníky a jedním studentem, který v této firmě taktéž vykonával bakalářkou praxi. K práci jsem používal vlastní notebook, ke kterému jsem dostal externí monitor, klávesnici a myš.

Mojí hlavní pracovní náplní bylo vytváření pluginů pro redakční systém Rocketoo. Abych se zorientoval ve fungování tohoto redakčního systému, byly mi ze začátku přiděleny jednoduché úkoly, jako například vygenerování jednoduchého XML feedu pro online srovnávač zboží, který obsahuje jednotlivé produkty a informace k nim. Dále jsem přešel k vytvoření složitějšího pluginu s připojením na databázi, až nakonec k implementaci platební brány.

3 Použité technologie

3.1 October CMS

October CMS je open-source platforma založená na nejpopulárnějším PHP frameworku Laravel. Slouží jako základ redakčního systému stránek, který se dá rozšířit o vlastní součásti, jako pluginy nebo témata vzhledu stránek.

3.1.1 Platforma PHP

October CMS je postaven na programovacím jazyce PHP, který byl poprvé vydán roku 1995 a používá se převážně k tvorbě dynamických webových stránek.[3] Hlavní vlastností tohoto jazyka je zpracování skriptu na straně serveru, díky čemu je uživateli odeslán pouze výsledek těchto skriptů.[2] Tímto je zajištěna také bezpečnost, jelikož uživatel k těmto skriptům nemá za normálních okolností přístup, na rozdíl od JavaScriptu, který se vykonává na straně uživatele.

3.1.2 Model-View-Controller

October CMS pracuje na architektuře Model-View-Controller. Tato architektura rozděluje software do tří na sobě nezávislých částí: [4]

- Model - Stará se o reprezentaci samotných informací, v našem případě dat z databáze. Jeden model zapouzdřuje přístup do jedné databázové tabulky.
- View - Slouží pro zobrazení výstupu uživateli, v našem případě se jedná o HTML stránky.
- Controller - Propojuje model a view. Využije model k načtení dat z databáze a zobrazí je pomocí view. Následně zaznamená akci uživatele na stránce (například kliknutí na tlačítko) a podle toho změní pomocí modelu data v databázi.

3.1.3 Pluginy

Pluginy jsou dle mého názoru jednou z nejlepších vlastností Octoberu. Pro pluginy má October CMS vyhrazenou složku *plugins*. V této složce má každý plugin svou složku, ve které se nachází veškerá logika a soubory týkající se daného pluginu. Základem každého pluginu je PHP třída *Plugin.php*, která se nachází v kořenovém adresáři pluginu a stará se o zaregistrování daného pluginu do systému, například aby se plugin zobrazil v menu a šlo na něj kliknout. Dále plugin obsahuje složky:

- assets - pro uložení souborů, jako například obrázků, které plugin využívá
- classes - pro uložení tříd
- controllers - pro uložení controllerů

- lang - pro uložení překladových souborů
- views - pro uložení HTML stránek zobrazovaných pluginem
- partials - pro uložení partials, což jsou části části HTML kódů, které se vkládají do jiných HTML stránek
- updates - pro uložení migrací, které popisuji v úkolu *Dárky k objednávce*

3.2 Nástroje pro správu verzování GitBucket a Source tree

GitBucket je verzovací systém, který jsem používal pro ukládání projektů. Verzovací systém obecně slouží pro práci v týmu na určitém projektu. Ukládá veškeré změny, které byly v projektu provedeny, díky čemu je možné vrátit projekt do jakékoliv předchozí verze, pokud je to potřeba. Verzovací systém navíc zabraňuje kolizím, což je situace, kdy dva uživatelé provedli změny ve stejném souboru, čímž by jeden uživatel přepsal změny uživatele druhého.

Pro práci s GitBucketem jsem používal aplikaci Source tree. Tato aplikace umožňuje v grafickém prostředí stahovat a nahrávat změny, prohlížet jednotlivé verze projektů včetně změn v kódu a podobně.

3.3 Editor Sublime Text

Sublime Text je velmi populární textový editor. Jeho velkou výhodou je možnost instalování balíčků neboli rozšíření, které editoru přidají nové funkce. Většina těchto balíčků je vytvořená komunitou a jsou zdarma k instalaci do Sublime Textu.[5] Příklady některých rozšíření, které jsem používal:

- GitGutter - zobrazí provedené změny v kódu
- Indent XML - umožňuje naformátovat XML soubor
- Random Everything - slouží pro generování náhodných čísel, jmen, adres a podobně
- SideBarEnhancements - přidá do kontextového menu boční lišty nové funkce, např.: nová složka, nový soubor, kopírovat, přejmenovat...

4 Náplň práce

V průběhu praxe jsem na základě požadavků od zákazníků řešil tyto úkoly:

- Generování XML feedu pro Pricemania.sk
- Cenový automat
- Dárky k objednávce
- Odběrná místa České pošty
- Platební brána GP webpay

5 Generování XML feedu pro Pricemania.sk

Pricemania.sk je slovenský online srovnávač zboží, který, podobně jako všechny ostatní srovnávače, porovnává nabídky jednotlivých obchodů u každého produktu. Zákazník si následně na těchto stránkách může vybrat pro něj nejvýhodnější nabídku na základě dostupnosti a ceny zboží nebo způsobu a ceny dopravy.

Mým úkolem bylo vygenerovat XML feed obsahující informace o všech aktuálně prodávaných produktech, díky kterému se na stránkách srovnávače zobrazí také nabídka z daného e-shopu. Dále bylo potřeba zajistit, aby se tento feed generoval a zobrazoval vždy po přístupu na určitou URL adresu. Díky tomu si srovnávač může kdykoliv stáhnout aktuální informace a zobrazit je na svých stránkách.

Pricemania.sk na svých stránkách poskytuje specifikaci struktury XML feedu, kterou vyžaduje. Podle této struktury bylo potřeba feed generovat, aby byl srovnávačem přijat a správně zpracován.

5.1 Postup řešení

Jako první krok jsem si pročetl specifikaci XML feedu, na kterou jsem dostal odkaz, abych měl představu o tom, jaký má takový feed pro srovnávač zboží tvar. S XML formátem jsem již dříve pracoval, díky čemu jsem se v jeho struktuře velmi rychle zorientoval. Přínosem taktéž bylo velmi kvalitní a přehledné zpracování specifikace, včetně příkladů správného a špatného použití jednotlivých tagů.

XML soubor má stejnou strukturu jako HTML. Skládá se z párových, libovolně pojmenovaných tagů, které obsahují konkrétní data nebo další tagy. Každý tag může navíc obsahovat atributy, které slouží pro konkrétnější specifikaci dat obsažených v daném tagu.

Jelikož tento plugin nebyl první, který v systému Rocketoo slouží primárně pro generování XML feedu, bylo potřeba dodržet stejnou strukturu, jako mají ostatní generátory feedů. Společná logika všech generátorů feedů je implementovaná v abstraktní třídě *FeedBase.php*. Tato třída obsahuje nejrůznější funkce pro usnadnění tvorby nového generátoru, například funkci pro nastavení URL adresy, na které bude feed dostupný.

Jedna z těchto funkcí slouží pro implementaci vlastní logiky, která naformátuje data z databáze do tvaru potřebného pro tento feed, jelikož určitá data byla v databázi uložena v jiném tvaru, než feed vyžadoval, nebo se v databázi neuchovávala vůbec. Pricemania například očekávala v XML feedu tag *availability*, který měl obsahovat dostupnost daného zboží ve dnech, zatímco v databázi e-shopu se dostupnost ukládala v hodinách. Jednoduchým výpočtem jsem převedl hodiny na dny a všechny informace o produktu jsem následně vypsal ve formátu XML.

Jelikož Pricemania.sk neposkytuje žádné testovací prostředí, ve kterém by bylo možno správnost XML feedu vyzkoušet, ani jsem neměl k dispozici žádný testovací účet, z mé strany hotový plugin jsem předal kolegovi, který se stará o back-end stránek, aby jej zkontroloval a případně mi jej vrátil na úpravu.

```xml
<products>
   <product>
      <id>7</id>
      <name>Samsung galaxy 1</name>
      <price>3000</price>
      <category>Mobilni telefony</category>
      <manufacturer>Samsung</manufacturer>
      <url>www.nazev-eshopu.cz/p/samsung-galaxy-1/</url>
      <shipping>94.00</shipping>
      <availability>0</availability>
      <params></params>
   </product>
</products>
```

Výpis 1: Ukázka výpisu XML feedu pro srovnávač zboží Pricemania.sk

5.2 Zhodnocení

Díky tomuto úkolu jsem se seznámil se základní strukturou pluginu v platformě October CMS. Celková doba práce na tomto úkolu byla přibližně 8 hodin včetně studování specifikace XML feedu. Plugin je momentálně k dispozici vlastníkům e-shopu, kteří chtějí tento srovnávač zboží využívat.

6 Cenový automat

Cenový automat je webová aplikace, která slouží k automatickému upravování cen na straně e-shopu pro zajištění vyšší návštěvnosti ze srovnávačů zboží. Funguje na principu, kdy Cenový automat zažádá e-shop o prodejní cenu určitého produktu, kterou porovná s prodejními cenami totožného produktu z jiných e-shopů. Cenový automat následně e-shopu odešle nově vypočtenou cenu produktu, díky které se e-shop v žebříčku e-shopů, nabízejících tento produkt na stránkách různých srovnávačů zboží, posune na vyšší příčky. Zákazníci jej díky tomu uvidí častěji a zvýší se tím návštěvnost e-shopu.

Mým úkolem bylo vytvořit plugin, který bude zajišťovat komunikaci mezi e-shopem a Cenovým automatem, včetně automatického nastavování nových cen na straně e-shopu. Druhým krokem pak bylo vytvoření nastavení pro tento plugin v administraci e-shopu.

6.1 Postup řešení

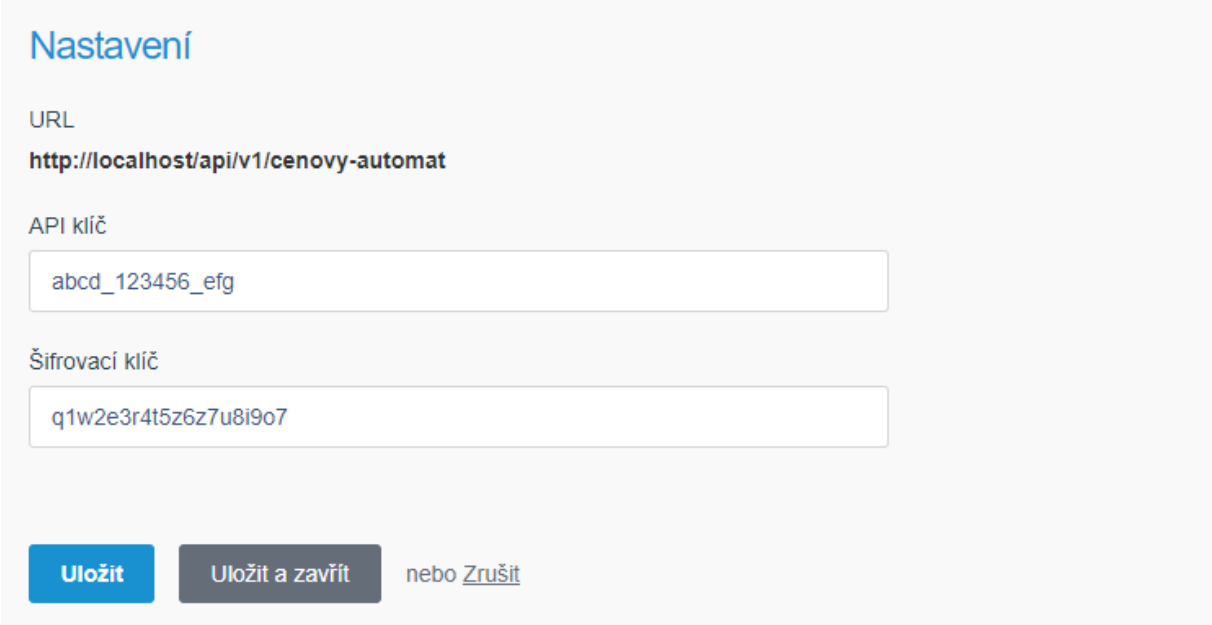
Před začátkem práce jsem od kolegy obdržel předpřipravený PHP script, který nám poskytl Cenový automat. Tento skript obsahuje veškerou logiku, která není závislá na struktuře systému, to znamená: přijetí dat z Cenového automatu, odesílání dat do Cenového automatu, šifrování dat, dešifrování dat a kontrola, jestli během přenosu nedošlo k podvržení dat. Kód v tomto scriptu je navíc velmi dobře okomentován, což značně usnadní a urychlí práci. Jediné dvě věci, které bylo potřeba doprogramovat, byly:

- Načtení ceny produktu z databáze
- Nastavení nové ceny produktu v databázi

```php
/* nacteni ceny produktu do promenne $price */
$variant = Variant::find($variandId);
if ($variant) {
   $price = $variant->price_purchase_vat;
}

/* nastaveni nove ceny produktu */
$variant = Variant::find($variantId);
if($variant){
   $variant->price_basic_vat = $newPrice;
   $variant->save();
}
```

Výpis 2: Ukázka načtení a nastavení ceny produktu



Obrázek 1: Nastavení cenového automatu

Chybějící funkcionalitu, kterou můžeme vidět ve výpisu, jsem do kódu doplnil a založil jsem nový plugin, do kterého jsem tento script implementoval. October CMS umožňuje každému pluginu vytvořit vlastní URL adresy, jenž můžou implementovat libovolnou funkcionalitu. Pro tento účel slouží soubor *routes.php*, který se nachází v kořenovém adresáři pluginu. V tomto souboru jsem vytvořil URL adresu, kterou bude Cenový automat využívat pro komunikaci s e-shopem, jelikož po přístupu na tuto adresu se script spustí a zpracuje požadavek Cenového automatu.

Cenový automat je placená služba, z tohoto důvodu je potřeba zajistit, aby jej mohl využívat pouze ten, kdo si jej zakoupí. Pro tento účel se používá takzvaný *apiKey*, který je jedinečný pro každého uživatele Cenového automatu. Tento klíč je ve tvaru textového řetězce skládajícího se z náhodných znaků a při komunikaci mezi e-shopem a automatem si jej automat ověří, aby věděl, se kterým uživatelem komunikuje a zda má službu zakoupenou. Druhým klíčem, který se využívá, je *cryptoKey*. Tento klíč má podobný tvar jako *apiKey* a slouží pro zašifrování a dešifrování veškerých dat, které se posílají mezi e-shopem a Cenovým automatem, aby se znemožnilo jejich přečtení a případnému upravení neoprávněnou osobou. Vytvořil jsem proto v administraci e-shopu nastavení, ve kterém má uživatel možnost tyto dva klíče, které mu vygeneruje Cenový automat při zakoupení jejich služby, nastavit. Při následném zavolání výše zmíněného scriptu se na určité místo v kódu klíče vloží, aby komunikace mezi e-shopem a automatem proběhla v pořádku.

Bohužel jsem neměl možnost provést test správného fungování tohoto pluginu. Neměl jsem k dispozici žádný testovací účet, který by měl přidělený apiKey a cryptoKey.

6.2 Zhodnocení

Naučil jsem se, jak v platformě October CMS vytvořit a používat nastavení pluginu. Práce na tomto úkolu trvaly přibližně 8 hodin. Plugin je momentálně k dispozici pro instalaci do e-shopu.

7 Dárky k objednávce

Dárky k objednávce byl můj třetí a zároveň první komplexnější úkol. Měl jsem za úkol vytvořit plugin, který bude sloužit pro vytváření dárků k objednávce, které zákazník dostane zdarma při nákupu nad určitou cenu. Tento plugin má za úkol umožnit administrátorovi e-shopu vytváření a úpravu jednotlivých dárků k objednávce, vypsat zákazníkovi v nákupním košíku seznam dárků, které zdarma obdrží a přidat dárky s nulovou cenou k objednávce při jejím vytvoření a uložení do databáze.

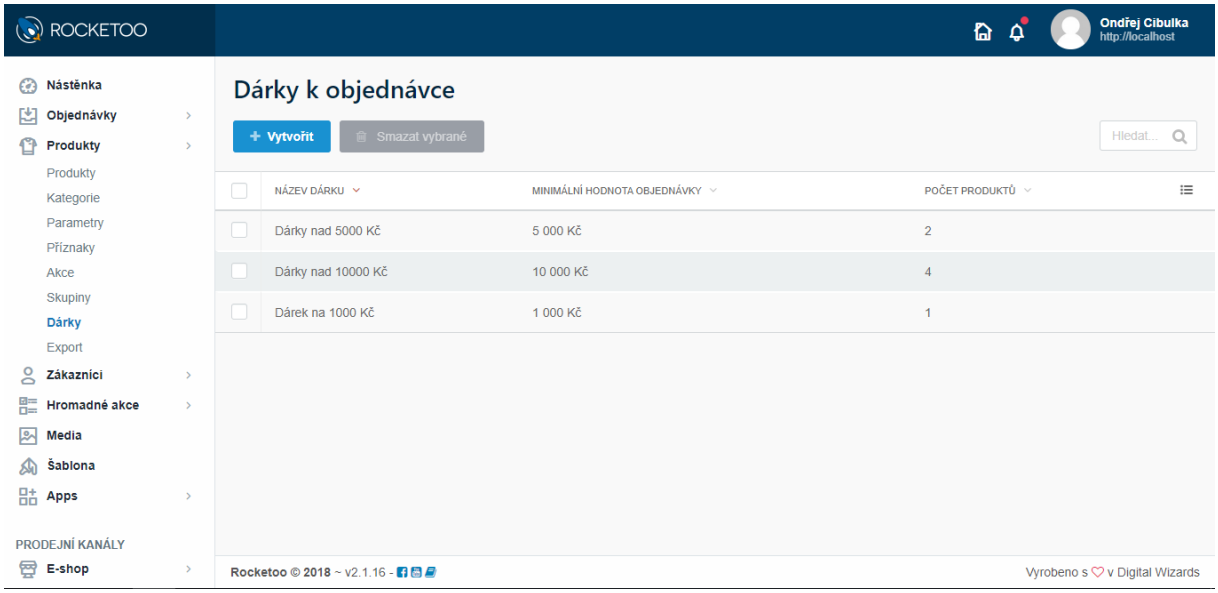
Dále bylo potřeba, aby každému dárku bylo možno nastavit název, který slouží jako interní pojmenování pro administrátora e-shopu, minimální hodnotu objednávky, ke které se dárky přidají a samotné dárky, které se vybírají z produktů nabízených e-shopem k prodeji, jelikož jeden dárek k objednávce může obsahovat několik produktů.

7.1 Postup řešení

Po založení nového pluginu jsem začal tvorbou administrátorské části, tedy možností vytvářet a upravovat jednotlivé dárky k objednávce v administraci e-shopu. Pro tento účel bylo potřeba navrhnout a vytvořit databázovou tabulku, která bude uchovávat veškeré informace o dárcích. Návrh databázové tabulky je potřeba důkladně promyslet, jelikož jakákoliv úprava se při již zprovozněném pluginu musí promítnout také na několika místech v kódu a hrozí, že se na jednu tuto úpravu zapomene, kvůli čemu nebude následně plugin fungovat správně. Také je potřeba vzít v potaz, jestli bude nutno v budoucnu plugin rozšířit o určitou funkčnost, která by zásadně ovlivnila strukturu databázové tabulky. S žádným rozšířením se ale nepočítalo, takže jsem navrhl jednoduchou tabulku, která obsahuje pouze údaje o dárku, tedy název a minimální cenu objednávky. Pro propojení dárků a produktů, které zákazník obdrží, bylo potřeba vytvořit propojovací tabulku, jelikož musela být použita vazba M:N, tedy jeden dárek k objednávce může obsahovat více produktů a jeden produkt se může nacházet ve více dárcích k objednávce a toto propojení nelze vytvořit bez pomoci propojovací tabulky.[6] October CMS využívá pro vytváření a úpravu databázových tabulek takzvané migrace. Migrace jsou soubory, které uchovávají strukturu databázové tabulky pomocí zdrojového kódu.

```
Schema::create('rocketoo_gifts_gifts', function($table)
{
   $table->engine = 'InnoDB';
   $table->increments('id')->unsigned();
   $table->string('name', 255);
   $table->decimal('price', 20, 2)->unsigned()->default(0);
}
```

Výpis 3: Ukázka migrace

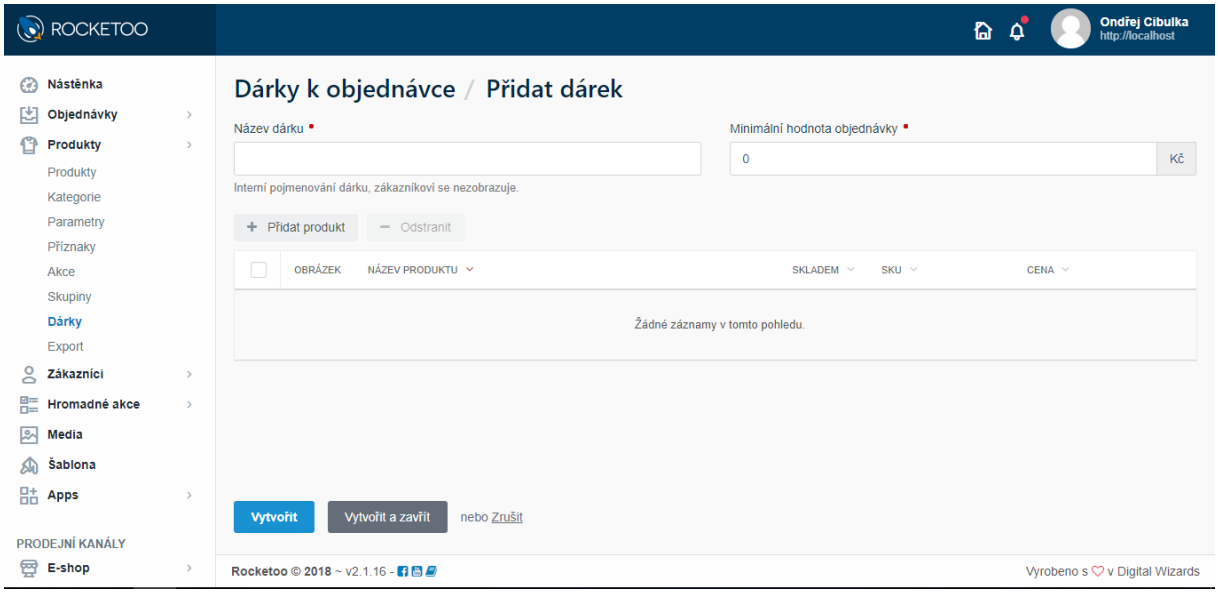


Obrázek 2: Výpis všech vytvořených dárků

Díky migracím je pak velmi snadné tabulky přenést a vytvořit je v databázi, ve které se zatím nenachází.

Administraci tohoto pluginu tvoří dvě stránky. První stránka zobrazuje administrátorovi seznam všech vytvořených dárků k objednávce. Administrátor má zde také možnost dárky odstranit, přejít na stránku pro úpravu konkrétního dárku, nebo na stránku pro vytvoření dárku nového. Druhá stránka slouží pro zmíněnou úpravu nebo tvorbu nového dárku. Zde má administrátor možnost nastavit všechny parametry dárku, to znamená: název, minimální cenu objednávky a konkrétní produkty, které zákazník zdarma obdrží.

Jelikož má tento plugin za úkol vypsat dárky do nákupního košíku, jehož kód se nachází ve zcela jiné části redakčního systému a není s mým pluginem nijak propojen, bylo potřeba tyto dvě na sobě nezávislé části propojit. Nežádoucí také bylo, aby byl košík s tímto pluginem propojen „napevno“. V praxi to znamená, aby se do kódu košíku nemusel připsat řádek zajišťující pouze vypsání dárků vždy, když se tento plugin do e-shopu nainstaluje. Pro tento účel se používají události, které lze na určitém místě v kódu vyvolat a na jiném místě pomocí posluchače zachytit. Nákupní košík při svém vykreslování vyvolává tři události podle místa, které se aktuálně vykresluje. Mě zajímala pouze událost s názvem *CART_MIDDLE_RENDER_ON*, která se vyvolává v momentě, kdy jsou v košíku již vypsány všechny produkty. V mém pluginu jsem vytvořil posluchače, který na tuto událost naslouchá a při jejím vyvolání spustí kód zajišťující vypsání dárků, na které má zákazník nárok. Jelikož události není nutno posluchačem zachytit a zpracovat, košík se bez chyby vykreslí i v případně, kdy nejsou dárky k objednávce v e-shopu nainstalovány. Další výhodou událostí je, že vyvolanou událost je možno zachytit několika posluchači, tudíž kterýkoliv plugin, který bude potřebovat do nákupního košíku pod výpis produktů něco vykreslit, může použít tutéž událost.



Obrázek 3: Vytvoření nového dárku

Jelikož mají události možnost přenášení dat, bylo ideální je použít i v případě přidání dárků k objednávce. V tomto případě bylo potřeba vědět, ke které objednávce v databázi se mají dárky přidat na rozdíl od košíku, který je uložen v *session*. Událost *ORDER_CREATE_AFTER*, která se vyvolává po vytvoření objednávky v databázi s sebou přenáší data, díky kterým je možno určit, o kterou objednávku se jedná. Podobně jako v předchozím případě, i tuto událost zachytává posluchač, který následně k dané objednávce přidá dárky, kterým se nastaví nulová cena a popis, že se jedná o dárek zdarma k objednávce.

Těsně před dokončením pluginu jsem dostal za úkol přidat nastavení, ve kterém si bude moct administrátor nastavit, jestli mají být dárky závislé na skladu. To znamená, aby se dárky k objednávce nepřidaly, pokud nebudou skladem. Toto jednoduché nastavení obsahuje pouze jeden přepínač, který umožní administrátorovi si tuto funkci zapnout. Následně jsem upravil logiku výběru dárků, která nyní při zapnuté závislosti na skladu ověří, jestli je počet jednotek skladem daného produktu větší než nula.

7.2 Zhodnocení

Díky tomuto úkolu jsem se seznámil se složitější strukturou pluginu v platformě October CMS. Naučil jsem se, jak plugin propojit s databází, jak fungují události a jak událost zachytit a zpracovat. Práce na tomto pluginu byla v rozsahu 8-10 dnů. Plugin je plně funkční a připraven pro instalaci do e-shopu.

8 Odběrná místa České pošty

Redakční systém Rocketoo umožňuje nainstalovat jako jednoho z mnoha dopravců Českou poštu. Česká pošta nabízí tři druhy přepravy balíků:

- Balík do ruky
- Balík na poštu
- Balík do balíkovny

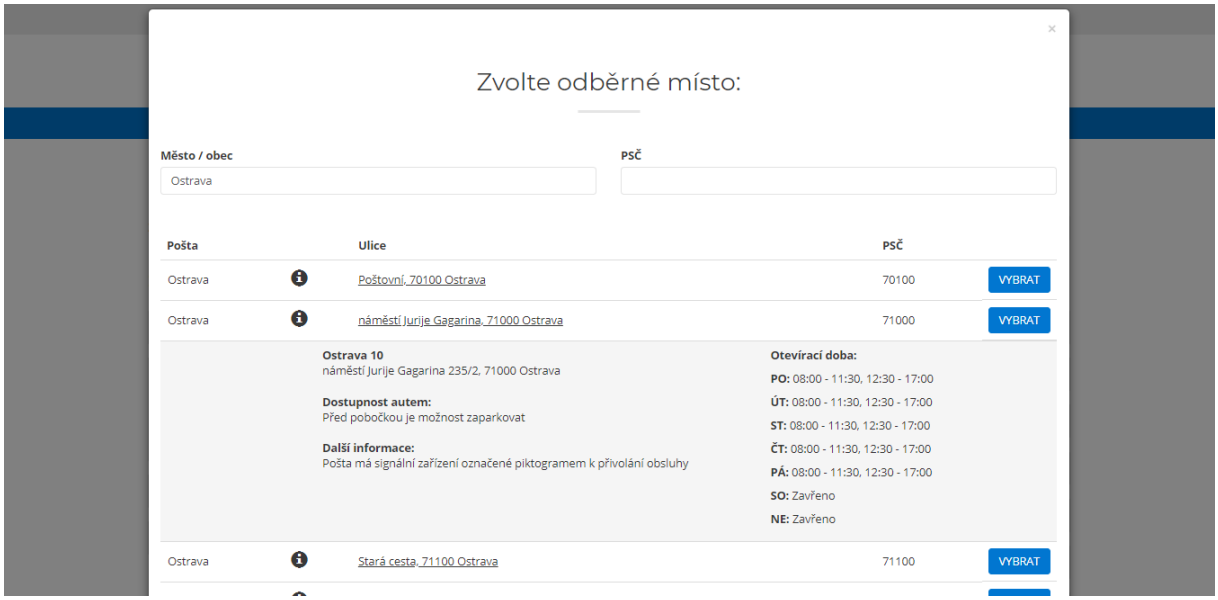
Mým úkolem bylo do tohoto dopravce doplnit funkcionalitu, která zákazníkovi při zvolení dopravy *Balík na poštu* umožní podle města, nebo poštovního směrovacího čísla vybrání pošty, na které si přeje balík vyzvednout. Tuto funkcionalitu měl již dopravce Zásilkovna, který taktéž umožňuje volbu odběrného místa. Bylo mi doporučeno se tímto pluginem inspirovat a dodržet podobnou strukturu a vzhled.

8.1 Postup řešení

Česká pošta poskytuje na svých webových stránkách JSON feed obsahující informace o všech poštách, na kterých je možno si balík vyzvednout. Tento feed je dostupný na adrese *https://b2c.cpost.cz/services/PostOfficeInformation/getDataAsJson* a je možno jej použít bez jakékoliv registrace. U každé pošty jsou přiloženy informace o adrese, souřadnicích, kontaktu, otevírací době a několika dalších možnostech pošty, které ale zákazník nakupující v e-shopu vědět nepotřebuje, takže jsem se jimi nezabýval.

Česká pošta, na rozdíl od Zásilkovny, umožňuje stahovat JSON feed v neomezené frekvenci. Jelikož odběrná místa různých dopravců se příliš často nemění, některé feedy lze stáhnout například pouze jednou za 24 hodin, aby se předešlo zbytečnému přetěžování serveru poskytující daný feed. V případě, že by frekvence stahováni byla Českou poštou omezená, musely by se odběrná místa uložit do databáze, ze které by se následně zobrazovaly zákazníkovi. Dále by se musela implementoval logika, která by například každých 24 hodin feed stáhla a informace aktualizovala v databázi. V našem případě ale Česká pošta stahování neomezuje, takže lze informace o poštách stáhnout vždy, když je bude potřeba zobrazit zákazníkovi.

Zákazník je před zobrazením konkrétních výdejních míst donucen zadat název města nebo poštovní směrovací číslo, podle kterých se zobrazí jen pošty nacházející se v dané lokalitě. Díky tomuto řešení nemusí zákazník prohledávat všechny existující pošty a urychlí se taktéž jejich vypsání na stránce. Podobně jako většina generátorů feedů, i tento umožňuje filtrovat výsledky podle určitých kritérii, mimo jiné také podle názvu města nebo poštovního směrovacího čísla. Filtry lze použít přidáním různých parametrů do URL adresy zobrazující tento feed. Pomocí pár podmínek v kódu jsem zjistil, podle kterého údaje si zákazník nechává vypsat odběrná místa a následně jsem vytvořil URL adresu se správným parametrem a hodnotou, aby odpovídala zákazníkovu vyhledávání.



Obrázek 4: Výběr odběrného místa České pošty

```
if ($city && $zip) {
            $url = 'https://b2c.cpost.cz/services/PostOfficeInformation/
                getDataAsJson?place='.urlencode($city).'&postCode='.$zip;
} elseif ($city) {
            $url = 'https://b2c.cpost.cz/services/PostOfficeInformation/
                getDataAsJson?place='.urlencode($city);
} elseif ($zip) {
            $url = 'https://b2c.cpost.cz/services/PostOfficeInformation/
                getDataAsJson?postCode='.$zip;
}
```

Výpis 4: Ukázka generování URL adresy

Podobně jako formát XML, tak i formát JSON je velmi rozšířený. V minulosti jsem s ním již mnohokrát pracoval a věděl jsem, jak jej pomocí jazyka PHP zpracovat a použít. Pro stažení feedu lze použít funkci *file_get_contents*, která jako první parametr očekává cestu k souboru, ze kterého chceme obsah načíst. Tato cesta nemusí být pouze URL adresa, ale například i cesta k uloženému souboru v počítači (serveru). Jelikož JSON je pouze textový řetězec, PHP z něj neumí načíst data nacházející se na určitém klíči. V tomto případě je potřeba použít funkci *json_decode*, která z JSON formátu vytvoří pole.

Posledním krokem bylo zformátování dat do tvaru příjemného pro zákazníka. Souřadnice pošty jsem použil pro vytvoření URL adresy na mapy Google, do které lze přidat atribut *q* ve tvaru *q=zeměpisnáŠířka,zeměpisnáDélka*. Tyto souřadnice se následně zákazníkovi na mapách

vyznačí špendlíkem, díky kterému se může jednoduše podívat, kde se pošta nachází. Náročnější bylo zformátovat otevírací hodiny. Každá pošta může mít v jeden den rozdělené otevírací hodiny až na tři úseky, mezi kterými byla pauza. Ověřil jsem, na kolik úseků je otevírací doba rozdělená a jednotlivé úseky jsem vypsal na jeden řádek oddělené čárkou. Pokud v určitý den není k dispozici žádná otevírací doba, vypíše se zákazníkovi k tomuto dni „Zavřeno“.

8.2 Zhodnocení

Prací na tomto úkolu jsem strávil přibližně 4 dny. Naučil jsem se, jak pomocí jazyka PHP stáhnout a využít feed poskytovaný jinou stránkou. Výběr odběrných míst je momentálně implementován v e-shopu BS Work na adrese *https://www.bswork.cz*.

9 Platební brána GP webpay

Dostal jsem za úkol vytvořit platební metodu, která bude umožňovat platbu na e-shopu pomocí platební brány GP webpay. Platební brána umožňuje zákazníkovi zaplatit kartou ihned po odeslání objednávky. Výhoda tohoto způsobu platby je, že platba je provedena okamžitě, tudíž e-shop může začít se zpracováním objednávky ihned a zboží bude zákazníkovi doručeno dříve.

Od kolegy jsem obdržel implementační manuál a přístupové údaje k účtu GP webpay, včetně testovací platební karty, abych mohl provádět testovací platby pro ověření správného fungování platební brány.

9.1 Postup řešení

Implementační manuál obsahoval kompletního popis platebního procesu. Tento manuál jsem si tedy nastudoval, abych se s fungováním platební brány seznámil. Princip platebních bran je vcelku jednoduchý a lze jej rozdělit na dvě části. První část je přesměrování zákazníka na adresu platební brány, kde zákazník platbu provede. Spolu s tímto přesměrováním se odešlou také data týkající se platby, jako například cena nákupu nebo email zákazníka. Druhá část je přijetí a zpracování výsledku platby na straně e-shopu a zobrazení výsledku zákazníkovi.

Před začátkem programování jakékoliv logiky jsem si z účtu GP webpay, ke kterému jsem dostal přístupové údaje, stáhl vzorovou implementaci této platební brány. Tato vzorová implementace je ve tvaru jednoduché webové stránky, kterou si je možno zprovoznit na svém počítači. Umožňuje přesměrování na platební bránu včetně údajů o platbě, které je možno si navolit. Před samotným přesměrováním zobrazuje přesnou URL adresu, včetně všech parametrů, na kterou bude zákazník přesměrován a po úspěšném zaplacení zobrazuje údaje odeslané zpět platební bránou.

Podobně, jako předchozí pluginy, ani tento nebyl první svého druhu. Platebních bran již bylo pro redakční systém Rocketoo naimplementováno mnoho, tudíž jsem se i zde držel předpřipraveného schématu. Začal jsem s implementací první části, tedy přesměrováním zákazníka na adresu platební brány. URL adresa, na kterou má být zákazník přesměrován, byla vypsána v implementačním manuálu podobně, jako povinné a volitelné parametry, které jsou potřeba k adrese přidat. Pro přesměrování byly k dispozici dvě URL adresy. První sloužila pro reálné zaplacení, druhá pro testovací účely. V obou těchto případech se používají stejné parametry a platební brána vrací stejné údaje, jediný rozdíl je v tom, že testovací platební brána neuskuteční žádnou reálnou platbu. Za pomoci implementačního manuálu a vzorové implementace jsem z databáze načetl všechny povinné údaje a vložil je jako parametry do URL adresy. Pro zajištění bezpečnosti se jako jeden z parametrů odesílá parametr signature, který má tvar dlouhého textového řetězce. Tento řetězec se generuje z ostatních odesílaných parametrů a pro zašifrování se používá certifikát, který lze stáhnout z účtu GP webpay. Logiku šifrování a dešifrování jsem ale řešit nemusel, jelikož GP webpay poskytuje ke stažení třídu napsanou v jazyce PHP, která tuto problematiku řeší.

Po funkčním přesměrování na stránku platební brány jsem se začal zabývat zpracováním výsledku platby, který platební brána odešle zpět e-shopu. Platba může skončit třemi stavy:

- Zaplaceno
- Zrušeno
- Chyba (například nedostatek financí na kartě)

Před samotným zpracováním výsledku platby bylo potřeba ověřit signature, který platební brána odesílá zpět spolu s výsledkem platby. K ověření bylo možné taktéž použít třídu poskytovanou společností GP webpay. Následně jsem podle stavu platby nastavil stav objednávky, díky kterému se administrátorovi e-shopu v administračním rozhraní objednávek zobrazí, že daná objednávka je zaplacena, nebo zrušena.

9.2 Zhodnocení

Díky tomuto pluginu jsem se naučil, na jakém principu funguje platební brána. Práce byly v rozsahu 8-10 dnů. Momentálně je tato platební brána připravená k použití v e-shopu.

10 Závěr

10.1 Znalosti získané v průběhu studia uplatněné v průběhu odborné praxe

Během své bakalářské praxe jsem nejvíce využil znalosti získané z předmětů zabývající se programováním. Konkrétně z předmětů *Programovací jazyky I* a *Programovací jazyky II*, kde se využívalo objektově orientované programování. Následně jsem využil znalosti tvorby databázových tabulek, jejich struktury a fungování získané v předmětu *Úvod do databázových systémů* a znalosti struktury softwaru získané v předmětu *Vývoj informačních systémů*.

10.2 Znalosti scházející v průběhu odborné praxe

Znalosti, které mi v průběhu praxe chyběly nejvíce, se týkaly struktury a fungování platformy October CMS a také již běžícího redakčního systému Rocketoo. Tyto znalosti ale postupem času vymizely, jelikož jsem při každém projektu řešil odlišný problém, který pracoval s jinými částmi systému.

Postrádal jsem také pokročilejší znalosti jazyka PHP, jako například práci s XML formátem nebo využití Traitu.

10.3 Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Během své bakalářské praxe jsem pracoval na vývoji pluginů pro redakční systém Rocketoo. Díky těmto úkolům jsem se zdokonalil v jazyce PHP, dozvěděl jsem se, jaké technologie můžou být použity při vývoji webových stránek a jak tyto technologie fungují. Naučil jsem se také, jak probíhá práce v týmu. Každý plugin, který jsem dostal za úkol vytvořit, jsem se snažil naprogramovat pomocí co nejlépe čitelného a na výkon nejméně náročného kódu. Pluginy jsou k dispozici administrátorům, kterým snad usnadní práci a dodají příjemnější zkušenost s používáním e-shopu.

Volbu bakalářské praxe hodnotím velmi kladně, protože věřím, že zkušenosti získané během mého působení ve firmě mi pomůžou se v budoucnu uplatnit na trhu práce.

Literatura

[1] Redakční systém Woodee2 CMS [online]. Ostrava: Digital Wizards Group, c2016 [cit. 2018-04-09]. Dostupné z: https://digitalwizards.cz/redakcni-system-woodee

[2] PHP. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2018 [cit. 2018-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/PHP

[3] MACH, Jakub. PHP pro úplné začátečníky. Praha: Computer Press, 2002. Rychle a jistě. ISBN 80-722-6633-0.

[4] Model-View-Controller. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2018-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/Model-view-controller

[5] Sublime Text. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2015 [cit. 2018-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/Sublime_Text

[6] STEPHENS, Ryan K., Ronald R. PLEW a Arie JONES. Naučte se SQL za 28 dní: [stačí hodina denně]. Brno: Computer Press, 2010. ISBN 978-80-251-2700-1.